-- --

IEEE P1003.2 Draft 11.2 − September 1991

Information technology — Portable Operating System Interface (POSIX) — Part 2: Shell and Utilities

Section 1: General

1.1 Scope

This standard defines a standard source code level interface to command interpretation, or ‘‘shell,’’ services and common utility programs for application programs. These services and programs are complementary to those specified by ISO/IEC 9945-1: 1990 {8}, hereinafter referred to as ‘‘POSIX.1 {8}.’’

The standard has been designed to be used by both application programmers and system implementors. However, it is intended to be a reference document and not a tutorial on the use of the services, the utilities, or the interrelationships between the utilities.

The emphasis of this standard is on the shell and utility functionality required by application programs (including ‘‘shell scripts’’) and not on the direct interactive use of the shell command language or the utilities by humans.

Portions of this standard comprise optional language bindings to system service interfaces. See, for example, the C Language Bindings Option in Annex B. This standard is intended to describe language interfaces and utilities in sufficient detail so that an application developer can understand the required interfaces without access to the source code of existing implementations on which they may be based. Therefore, it does not attempt to describe the source programming language or internal design of the utilities; they should be considered ‘‘black boxes’’ that exhibit the described functionality.

For language interfaces, or functions, this standard has been defined exclusively at the source code level. The objective is that a conforming portable application source program can be translated to execute on a conforming implementation. The standard assumes that the source program may need to be retranslated to produce target code for a new environment prior to execution in that environment.

There is no requirement that the base operating system supporting the shell and utilities be one that fully conforms to ISO/IEC 9945-1: 1990 {8}. (The base system could contain a subset of POSIX.1 {8} functionality, enough to support the requirements for this standard, as described in 2.9.1, but that could not claim full conformance to all of POSIX.1 {8}.) Furthermore, there is no requirement that the shell command interpreter or any of the standard utilities be written as POSIX.1 {8} conforming programs, or be written in any particular language.

Although not requiring a fully conforming POSIX.1 {8} base, this standard is based upon documentation and the knowledge of existing programs that assume an interface and architecture similar to that described by POSIX.1 {8}. Any questions regarding the definition of terms or the semantics of an underlying concept should be referred to POSIX.1 {8}.

1.1.1 Scope Rationale. *(This subclause is not a part of P1003.2)*

This standard is one of a family of related standards. The term POSIX is correctly used to describe this family, and not only its foundation, the operating system interfaces of POSIX.1 {8}. Therefore, POSIX.2 could colloquially be described as the 'POSIX Shell and Tools Standard.''

The interfaces documented for this standard are to and from high-level language application programs and to and from the utilities themselves; the standard does not directly address the interface with users.

The ''source code'' interface to the command interpreter is defined in terms of high-level language functions in 7.1.1 or 7.1.2 (such as *system* (), B.3.1, or *popen* (), B.3.2). There are also other function interfaces, such as those for matching regular expressions in 7.3 (*regcomp* () in B.5). Many of the utilities in this standard, and the shell itself, also accept their own command languages or complex directives as input data, which is also referred to as source code. This data, an ordered series of characters, may be stored in files, or ''scripts,'' that are portable between systems without true recompilation. However, just as with POSIX.1 {8}, the standard addresses only the issue of source code portability between systems; applications using these calls may have to be recompiled or translated when moving from one system to another.

There has been considerable debate concerning the appropriate scope of the work represented by this standard. The following are rational alternatives that have been evaluated:

(1) Define the shell and tools as extensions to POSIX.1 {8}. This would require a full conforming POSIX.1 {8} system as a base for the new facilities described here. Vocal proponents for this view have been the members of the POSIX.3 working group, who foresaw difficulties in

producing a verification suite standard without having a known operating system base.

(2) Decouple the shell and tools entirely from POSIX.1 {8}. This would potentially allow the standard to be implemented on such popular operating systems as MVS/TSO, VM/CMS, MS/DOS, VMS, etc. Those systems would not have to provide every minor detail of the POSIX.1 {8} language interfaces to conform under this model—only enough to support the shell and tools.

(3) Compromise between options 1 and 2. Base the standard on an interface *similar* to POSIX.1 {8}, but don't require full conformance. A simple example would be a Version 7 UNIX System, which could not conform to POSIX.1 {8} without considerable modification. However, a vendor could support all of the features of this standard without changing its kernel or binary compatibility. Another example would be a system that conformed to all stated POSIX.1 {8} interfaces, but that didn't have a fully conforming C Standard {7} compiler. The difficulty with this option is that it makes the stated goal of the working group a bit fuzzier and increases the amount of analysis required for the features included.

The working group selected option 3 as its goal. It chose to retain the full UNIX system-like orientation, but did not wish to arbitrarily deprive legitimate systems that could *almost* conform. No useful feature of shells or commonly-used utilities were discarded to accommodate nonconforming base systems; on the other hand, no deliberate obstacles were arbitrarily erected. Furthermore, POSIX.1 {8} is still required for its definitions and architectural concepts, which are purposely not repeated in this standard.

One concrete example of how the two standards interrelate is in the usage of POSIX.1 {8} function names in the descriptions of utilities in POSIX.2. There are a number of historical commands that directly mapped into one of the UNIX system calls. For example: `chmod` and *chmod*(); `ln` and *link*(). The POSIX.2 working group was faced with the problem of having to define all of the complex interactions ''behind the scenes'' for some simple commands. Creating a file, for example, involves many POSIX.1 {8} concepts, including processes, user IDs, multiple group permissions (which are optional), error conditions, etc. Rather than enumerating all of these interactions in many places, the POSIX.2 group chose to employ the POSIX.1 {8} function descriptions, where appropriate. See the `chmod` utility in 4.7 as an example. The utility description includes the phrase:

> . . . performing actions equivalent to the *chmod*() function as defined in the POSIX.1 {8} *chmod*() function:

This means that the POSIX.2 implementor has to read the POSIX.1 {8} *chmod*() description and fully understand all of its functionality, requirements, and side effects, which now don't have to be repeated here. (Admittedly, this makes the POSIX.2 standard a bit more difficult to read, but the working group felt that precision transcended the need for readable or semi-tutorial documents.)

The Introduction states that one of the goals of the working group was: ''This interface should be implementable on conforming POSIX.1 {8} systems.'' This implies that the working group has attempted to ensure that no additional functionality or extension is required to implement this standard on the base defined by POSIX.1 {8}. This is not to say that extensions are not allowed, but that they should not be necessary. The goal ‘‘(7) Utilities and standards for the installation of applications" was once interpreted to mean that an elaborate series of tools was required to install and remove applications, based on complex description files and system databases of capabilities. An attempt to provide this was rejected by the balloting group and that type of system is now being evaluated by the POSIX.7 System Administration group. However, the original goal remains in the list, because many of the standard utilities are, in fact, targeted specifically for application installation— `make`, `c89`, `lex`, etc.

1.1.1.1 Existing Practice. *(This subclause is not a part of P1003.2)*

The working group would have been very happy to develop a standard that allowed all historical implementations (i.e., those existing prior to the time of publication) to be fully conforming and all historical applications to be Strictly Conforming POSIX Shell Applications without requiring any changes. Some modifications will be required to reconcile the specific differences between historical implementations; there are many divergent versions of UNIX systems extant and applications have sometimes been written to take advantage of features (or bugs) on specific systems. Therefore, the working group established a set of goals to maximize the value of the standard it eventually produced. These goals are enumerated in the following subclauses. They are listed in approximate priority sequence, where the first subclause is the most important portability goal.

1.1.1.1.1 Preserve Historical Applications

The most important priority was to ensure that historical applications continued to operate on conforming implementations. This required the selection of many utilities and features from the most prevalent historical implementations. The working group is relying on the following factors:

(1) Many inconsistent historical features will still be supported as *obsolescent* .

(2) Common features of System V and BSD will continue to be supported by their sponsors, even if they aren’t included here (just as long as they are not prevented from existing).

Therefore, the standard was written so that the large majority of well-written historical applications should continue to operate as Conforming POSIX Shell Applications Using Extensions.

1.1.1.1.2 Clean Up the Interfaces

The working group chose to extend the benefits of historical UNIX systems by making limited improvements to the utility interfaces; numerous complaints have been heard over the years about the inconsistencies in the command line interface, which have allegedly made it harder for novice users. Given the constraints of **Preserve Historical Applications**, the working group has made the following general modifications:

(1) Utilities have been extended to deal with differences in character sets, collating sequences, and some cultural aspects relating to the locale of the user. (Examples: new features in regular expressions; new formatting options in `date`; see 4.15.)

(2) The utility syntax guidelines in 2.10.2 have been applied to almost all of the utilities to promote a consistent interface. The guidelines themselves have been loosened up a bit from their counterparts in the *SVID*. In many cases historical utilities have not conformed with these guidelines (which were written considerably later than the utilities themselves). The older interfaces have been maintained in the standard as obsolescent features. (Examples: `join`, `sort`.) However, in some cases, such as `dd` and `find`, such major surgery was required that the working group decided to leave the historical interfaces as is. ‘‘Fixing’’ the interface would mean replacing the command, which would not help applications portability. So, fixing was limited to relatively minor abuses of the new guidelines, where reasonable consistency could be achieved while still maintaining the general type of interface of the historical version.

(3) Features that were not generally portable across machine architectures or systems have been removed or marked obsolescent and new, more portable interfaces have been introduced. (Examples: the octal number methods of describing file modes in `chmod` and other utilities have been marked obsolescent; the symbolic ‘`ugo`’ method has been extended to other utilities, such as `umask`.)

(4) Features that have proved to be popular in some specific UNIX system variants have been adopted. (Examples: `diff −c`, which originated in BSD systems, and the ‘‘new’’ `awk`, from System V.) Such features were selected given the requirements for balloting group consensus; the features had to be used widely enough to balance accusations of ‘‘creeping featurism’’ and violations of the UNIX system ‘‘tools philosophy.’’

(5) Unreasonable inconsistencies between otherwise similar interfaces have been reconciled. (Example: methods of specifying the patterns to the three `grep`-related utilities have been made more consistent in the standard’s single `grep`.)

(6) When irreconcilable differences arose between versions of historical utilities, new interfaces (utility names or syntax) were sometimes added in their places. The working group resisted the urge to deviate significantly from historical practice; the new interfaces are generally consistent with the philosophy of historical systems and represent comparable functionality to the interfaces being replaced. In some cases, System V and BSD had diverged (such as with `echo` and `sum`) so significantly that no compromises for a common interface were possible. In these cases, either the divergent features were omitted or an entirely new command name was selected (such as with `printf` and `cksum`).

(7) Arbitrary limits to utility operations have been removed. (Example: some historical `ed` utilities have very limited capabilities for dealing with large files or long input lines.)

(8) Arbitrary limitations on historical extensions have been eliminated. (Example: regular expressions have been described so that the popular `\<` . . . `\>` extension is allowed.)

(9) Input and output formats have been specified in more detail than historical implementations have required, allowing applications to more effectively operate in pipelines with these utilities. (Example: `comm`.)

Thus, in many cases the working group could be accused of ‘‘violating Existing Practice,’’ and in fact received some balloting objections to that effect from implementors (although rarely from users or application developers). The working group was sensitive to charges that it was engaged in arbitrary software engineering rather than merely codifying existing practice. When changes were made, they were always written to preserve historical applications, but to move new conforming applications into a more consistent, portable environment. This strategy obviously requires changes to historical implementations; the working group carefully evaluated each change, weighing the value to users against the one-time costs of adding the new interfaces (and of possibly breaking applications that took advantage of bugs), generally siding with the users when the costs to implementations and applications was not excessively high.

In some cases, changes were reluctantly made that could conceivably break some historical applications; the working group allowed these only in the face of practices it considered rare or significantly misguided.

1.1.1.1.3 Allow Historical Conforming Applications

It is likely that many historical shell scripts will be Strictly Conforming POSIX.2 Applications without requiring modifications. Developers have long been aware of the differences among the historical UNIX system variants and have avoided the nonportable aspects to increase the scope of their applications' marketplace. However, the previous goal of a consistent interface was considered to be quite important, so there will be modifications required to some applications if they wish to be maximally portable in the future.

1.1.1.1.4 Preserve Historical Implementations

As explained in 1.1.1.1.2, the requirements for portability and a consistent interface have caused the working group to add new utilities and features. No historical implementations contained all of the attributes required by the working group. Therefore, this lowest priority goal fell victim to the preceding goals, and every known historical implementation will require some modifications to conform to this standard.

The working group took care to ensure that the implementations could add the new or modified features without breaking the operation of existing applications. (Note that the standard utilities are not considered applications in this regard, but are part of the implementation. In fact, many or most of the utilities named by this standard will have to change to some extent.)

1.1.1.2 Outside the Scope. *(This subclause is not a part of P1003.2)*

The following areas are outside the scope of this standard. This subclause explains more of the rationale behind the exclusions. (It should be noted that this is not an official list. It was not part of the Project Authorization Request submitted to the IEEE , but was devised as a guide to keep the working group discussions on track.)

(1) *Operating system administrative commands (privileged processes, system processes, daemons, etc.).*

The working group followed the lead of the POSIX .1 {8} group in this instance. Administrative commands were felt to be too implementation dependent and not useful for application portability. Subsequent to this decision, a separate POSIX. 7 working group was formed to deal with this area of ‘‘operator portability.’’ It is anticipated that utilities needed for system administration will be closely coordinated with the POSIX. 2 working group.

(2) *Commands required for the installation, configuration, or maintenance of operating systems or file systems.*

This area is similar to item (1). System installation is contrasted against the application installation portion of the Scope by its orientation to installing the operating system itself, versus application programs. The exclusion of operating system installation facilities should not be interpreted to mean that the application installation procedures *cannot* be used for installing operating system components. The proposed interface for this area encountered stiff resistance from the balloting group in Draft 8 and was temporarily withdrawn. As described in Annex E.4, a decision of the balloting group is pending on whether to begin work on a supplement to this standard (POSIX. 2b) for application installation.

(3) *Networking commands.*

These were excluded because they are deeply involved with other standards making bodies and are probably too complicated. In this case, several working groups were formed within the POSIX family to deal with this. It is anticipated that utilities needed for networking, if any, will be closely coordinated with the POSIX. 2 working group. (In early drafts of this standard, which predated the formation of the networking-specific POSIX working groups, the historical ‘UNIX system to UNIX system copy [UUCP]’’ programs and protocols were included. These descriptions have been removed in deference to a more appropriate working group.)

(4) *Terminal control or user-interface programs (e.g., visual shells, visual editors, window managers, command history mechanisms, etc.).*

This is probably the most contentious exclusion. A common complaint about many UNIX systems is how they're not very ''user friendly.'' Some people have hoped that the interface to users could be standardized with mice, icon-based desktop metaphors, and so forth. This standard neatly sidesteps those concerns by reminding its audience that it is an application portability standard, and therefore has little relationship to the manner in which users manage their terminals.

However, this guideline was not meant to apply to applications. It is perfectly reasonable for an application to assume it can have a user interacting with it. That is why such facilities as displaying strings (with `printf`) without `<newline>`s, `stty`, and various prompting utilities are included in the standard.

The interfaces in this standard are very oriented to command lines being issued by shell scripts, or through the *system* () or *popen* () functions. Therefore, interactive text editors, pagers, and other user interface tools have been omitted for now. Alternatively, other standards bodies, such as X3H3.6 and the IEEE TCOS P1201 working group, are devising interfaces that could possibly be more useful and long-lived than any prescribed by POSIX. 2.

There is one area of this subject that will be addressed by POSIX. 2. The scope of the working group has been expanded to include what is being termed the *User Portability Extension* , POSIX. 2a. This will be published as a supplement to this standard and have the goal of providing a portable environment for relatively expert time-sharing or software development users. It will not attempt to deal with mice or windows or other advanced interfaces at this time, but should cover many of the terminal-oriented utilities, such as a full-screen editor, currently avoided by this edition of POSIX. 2.

(5) *Graphics programs or interfaces.*

See the comments on user interface, above.

(6) *Text formatting programs or languages.*

The existing text formatting languages are generally too primitive in scope to satisfy many users, who have relied on a myriad of macro languages. There is an ISO standard text description language, SGML , but this has had insufficient exposure to the UNIX system community for standardization as part of POSIX at this time.

(7) *Database programs or interfaces (e.g. SQL , etc.).*

These interfaces are the province of other standards bodies.

1.1.1.3 Language-Independent Descriptions. *(This subclause is not a part of P1003.2)*

The POSIX.1 {8} and POSIX.5 working groups are currently engaged in developing the model for language-independent descriptions of system services. When complete, it will allow the C language bias of the POSIX.1 {8} standard to be excised and C will take its place among other language bindings that interface with the core services descriptions. The POSIX.2 working group did not wish to duplicate effort, and has therefore waited until POSIX.1 {8} achieves progress in this area. Thus, like the first version of POSIX.1 {8}, the initial drafts of POSIX.2 start life as a C-only standard, with language independence scheduled to be included in a later draft. Fortunately, this standard is substantially less involved with C than POSIX.1 {8} is. In fact, all of the C interfaces are entirely optional.

1.1.1.4 Base Documents. *(This subclause is not a part of P1003.2)*

The working group consulted a number of documents in the course of its deliberations, to select utilities and features. There were five primary documents that started off the process:

(1) The *System V Interface Definition* (*SVID*), Issue 2, Volume 2.

(2) The *X/Open Portability Guide*, (*XPG*), Issues II and III, Volume 1.

(3) *The UNIX User's Reference Manual*, 4.3 Berkeley Software Distribution, Virtual VAX-11 Version. (The printed documentation as well as the online versions provided with the BSD ''Tahoe'' and ''Reno'' distributions were considered as one base document for the POSIX.2 work.)

(4) *The KornShell Command and Programming Language*, by Bolsky and Korn.

(5) *The AWK Programming Language*, by Aho, Kernighan, and Weinberger.

The *XPG* was used most heavily in initial deliberations about which utilities and features to include. The X/Open companies had done a very thorough job in analyzing the *SVID* and other standards to compile a list of the most useful and portable utilities. They carefully marked many features that had portability problems and the working group avoided them for this standard.

AT&T, X/Open, and Berkeley provided machine-readable documentation for the use of the working group. However, due to very substantial differences in formatting standards, there is little resemblance between some of the utilities described here and their cousins in the *SVID*, *XPG*, and BSD user manual. Nevertheless, early usage of these documents was an invaluable aid in the production of the standard and the POSIX.2 working group extends its sincere thanks to all three organizations for their generous cooperation.

The biggest divergence in POSIX.2's documentation has been its philosophy of fully specifying interfaces. The *SVID* and *XPG* are oriented solely towards application portability. Implementors would have a difficult time writing some of these utilities from the descriptions alone. In fact, both documents freely rely on the potential implementors licensing the source code for the reference systems to complete the specification. The POSIX.2 standard, on the other hand, also has implementors in its audience and it strove to expand its descriptions wherever useful and feasible. For example, it makes use of BNF grammars to describe complex syntaxes. It attempts to describe the interactions between options, operands, and environment variables, where conflicts can exist. It also attempts to describe all of the useful utility input and output formats. The goal here was to allow application developers to write filters or other programs that could parse the output of any of these utilities or to provide meaningful input from their programs. To the working group's knowledge, this is a task never before attempted for the historical UNIX system commands—the source code was always so readily available to anyone who really needed to know this information.

The two commercial books listed were used as reference materials in preparing information on the shell and the *awk* language that was more recent and complete than AT&T's or X/Open's documentation.

1.1.1.5 History. *(This subclause is not a part of P1003.2)*

The *1984 /usr/group Standard* was originally intended to include the shell and user level commands. However, the /usr/group (now known as ‘‘UniForum’’) Standards Committee was unable to begin this effort, due to the complexity of the system call and library functions that it eventually did publish.

A shell was referred to in the *system* () function defined by *ANSI /X3.159-1989 Programming Language C Standard*, but no syntax for the shell command language was attempted.

As the first version of POSIX.1 {8} neared completion, it became apparent that the usefulness of POSIX would be diminished if no shell or utilities were defined. Therefore, the POSIX.2 working group was formed in January 1986 at the Denver, Colorado, meeting of POSIX.1 {8} to address this concern.

The progress of the working group has seemed rather slow during the more than three years of its existence. This is primarily because its membership had substantial overlap with the POSIX.1 {8} working group; for example, the Chair of POSIX.2 was also the Technical Editor of POSIX.1 {8} (and POSIX.2 as well!) at the time. And, meetings were arbitrarily shortened to allow the POSIX.1 {8} group to move forward as quickly as possible.

1.1.1.6 Internationalization. *(This subclause is not a part of P1003.2)*

Some of the utilities and concepts described in this standard contain requirements that standardize multilingual and multicultural support. Most of the internationalized support for this standard was proposed by the UniForum Technical Committee Subcommittee on Internationalization, at the request of the POSIX.2 working group.

UniForum, a nonprofit organization, organizes subcommittees of Technical Committees to do standards research on different topics pertinent to POSIX. The

UniForum Subcommittee on Internationalization is one such group. It was formed to propose and promote standard internationalized extensions to POSIX-based systems. The POSIX.2 working group and the UniForum Subcommittee on Internationalization coordinated their work by the use of liaison members, who attended the meetings of both groups. The interaction between the two groups started when POSIX.2 asked the Subcommittee on Internationalization to provide internationalized support for regular expressions. Later, the Subcommittee on Internationalization was charged with identifying areas in the standard needing changes for internationalized support and proposing those changes.

1.1.1.7 Test Methods. *(This subclause is not a part of P1003.2)*

The POSIX.3 working group has worked on a test methods specification for verifying conformance to POSIX standards in general and POSIX.1 {8} and POSIX.2 in particular. Test methods for POSIX.2 should be published as a separate document [1] sometime after POSIX.2 is approved.

1.1.1.8 Organization of the Standard. *(This subclause is not a part of P1003.2)*

The standard document is organized into sections. Some of these, such as the Scope in 1.1, are mandated by ISO/IEC, the IEEE, and other standards bodies. The remainder of the document is organized into small sections for the convenience of the working group and others. It has been suggested that all of the utility descriptions (and maybe the functions, too) should be lumped into one large section, all in alphabetical order. This would presumably make it easier for some users to use the document as a reference document. The working group deliberately chose to not organize it in this way, for the following reasons:

(1) Certain sections are optional. It is more convenient for the document's internal references, and also for people specifying systems, if these optional sections are in large pieces, rather than a detailed list of utility names.

(2) Future supplements to this standard will be adding new utilities that will also be optional. It would be confusing to try to merge documents at a level below major sections (chapters).

---

1) See the Foreword for information on the activities of other POSIX working groups.

1.2 Normative References

The following standards contain provisions which, through references in this text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of this International Standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

{1} ISO/IEC 646: 1983, [2] *Information processing— ISO 7-bit coded character set for information interchange.*

{2} ISO 1539: 1980, *Programming languages— FORTRAN.*

{3} ISO 4217: 1987, *Codes for the representation of currencies and funds.*

{4} ISO 4873: 1986, *Information processing— ISO 8-bit code for information interchange—Structure and rule for implementation.*

{5} ISO 8859-1: 1987, *Information processing—8-bit single-byte coded graphic character sets—Part 1: Latin alphabet No. 1.*

{6} ISO 8859-2: 1987, *Information processing—8-bit single-byte coded graphic character sets—Part 2: Latin alphabet No. 2.*

{7} ISO/IEC 9899: 1990, *Information processing systems—Programming languages—C.* 1

{8} ISO/IEC 9945-1: 1990, *Information technology—Portable Operating System Interface (POSIX)—Part 1: System Application Program Interface (API) [C Language]*

1.3 Conformance

1.3.1 Implementation Conformance

1.3.1.1 Requirements

A *conforming implementation* shall meet all of the following criteria:

---

2) Under revision. (This notation is meant to explicitly reference the 1990 Draft International Standard version of ISO/IEC 646.)

ISO/IEC documents can be obtained from the ISO office, 1, rue de Varembé, Case Postale 56, CH-1211, Genève 20, Switzerland/Suisse.